

Multi-borders classification

Peter Mills
 Peteysoft Foundation
petey@peteysoft.org

May 20, 2014

Abstract

The number of possible methods of generalizing binary classification to multi-class classification increases exponentially with the number of class labels. Often, the best method of doing so will be highly problem dependent. Here we present classification software in which the partitioning of multi-class classification problems into binary classification problems is specified using a recursive control language.

1 Introduction

Many statistical classification methods distinguish between only two classes by drawing a hyper-surface in the feature space. In a Support Vector Machine (SVM), the hyper-surface is drawn by minimizing the classification error. By implicitly transforming the feature space through operations on the dot product, the shape of this hyper-surface can be made quite complex (Müller et al., 2001). In Mills (2011) the hyper-surface is discretely sampled by finding the root of the difference in conditional probabilities along a series of lines drawn between the two classes. The conditional probabilities are found using a kernel density estimation technique (Terrell and Scott, 1992) called Adaptive Gaussian Filtering (AGF).

There are many methods of generalizing binary classification schemes to more than two classes. The LIBSVM library (Chang and Lin, 2011), for instance, uses a “one-against-one” approach wherein each class is compared against every other class. For large numbers of classes this approach is quite inefficient since there will be $n_c(n_c - 1)/2$ binary classifications, where n_c is the number of classes. Many other methods exist and the possible number will increase exponentially with the number of classes.

In many problems a different method of dividing or partitioning the classes would be appropriate. Consider four land surface types: coniferous forest, deciduous forest, corn field and wheat field. Here a hierarchical scheme (also called a decision tree) seems most appropriate since the related surface types will cluster together: first discriminate between forest and field. If forest is returned, then discriminate between evergreens and hardwoods. If field, then between corn and wheat. As another example, in a classification problem involving discretized continuum values, it makes sense to place the partitions between classes that define adjacent ranges in the continuum data.

New extensions to the *libAGF* library (Mills, 2011) generalize the binary classification problem so that the most appropriate method can be used to partition a multi-class problem without having to write a new program in each case. The AGF borders-training method has been paired with this algorithm, the combination of which we refer to as “multi-borders”. In what follows, we describe the rationale behind the software, how it works and test it on an example problem comprised of discretized continuous data.

2 Theory

Suppose we have several partitions as in Figure 1(a), each uniquely grouping all the classes into two sets. The following equations relate the conditional probabilities of the classes to those returned by the binary partitions:

$$\sum_{j=1}^{n_c} P(j|\vec{x}) = 1$$

$$\sum_{j=1}^{n_{2i}} P(c_{2ij}|\vec{x}) - \sum_{j=1}^{n_{1i}} P(c_{1ij}|\vec{x}) = P_i(2|\vec{x}) - P_i(1|\vec{x}),$$

where $P(c|\vec{x})$ is the conditional probability of class c at test point \vec{x} , $P_i(1|\vec{x})$ and $P_i(2|\vec{x})$ are the conditional probabilities of the first and second classes respectively on either side of partition i . The classes contained in either side of the i th partition are given by $\{c_{1ij}|j = [1, n_{1i}]\}$ and $\{c_{2ij}|j = [1, n_{2i}]\}$, respectively.

We call this *non-hierarchical* multi-borders classification. The popular “one-against-the-rest” approach, in which each class is singled out and classified against the remaining is one example of non-hierarchical classification and will be over-determined for every case. Note that the one-against-one

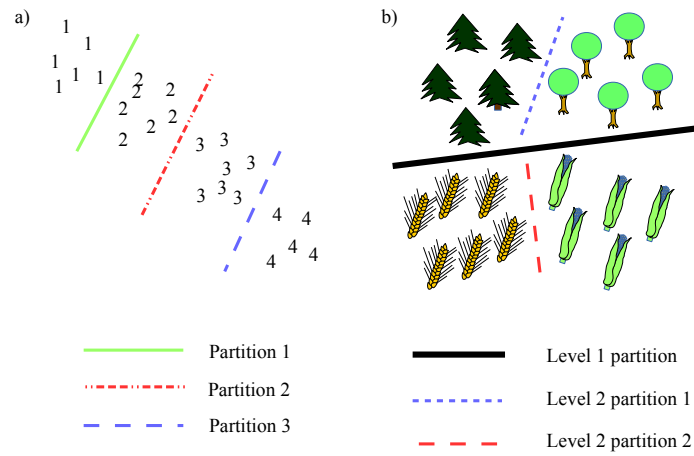


Figure 1: Comparison between non-hierarchical (a) and hierarchical classification (b).

approach is not covered by this method nor by the hierarchical approach described below since it requires excluding data from certain classes in absence of any prior knowledge of the class of the test point.

In a *hierarchical* classification scheme (or decision tree), the classes are first partitioned, then each of those partitions are partitioned again and so on until a class number is returned instead of another partition. The scenario for the first example is illustrated in Figure 1(b). A big difference between this and the non-hierarchical approach, is that all data from classes in the losing partition are excluded from subsequent classifications, whereas in the non-hierarchical approach all the data is used in all the binary classifications. As a corollary, in hierarchical multi-borders classification, only the conditional probability of the winning class is returned, whereas the non-hierarchical method solves for all of them. The two types can of course be combined.

3 Control language

A recursive control language is used to describe any possible configuration in this hierarchical approach. In Backus-Naur form, the control language looks like this:

```

<branch>      ::= <model> "{" <branch-list> "}" | <CLASS>
<model>       ::= <TWOCCLASS> | <partition-list>
<branch-list> ::= <branch> | <branch-list> <branch>
<partition-list> ::= <partition> | <partition-list> <partition>
<partition>   ::= <TWOCCLASS> <class-list> " / " <class-list> ";"
<class-list>  ::= <CLASS> | <class-list> " " <CLASS>

```

<CLASS> is a class value between 0 and $n_c - 1$. It is used in two senses. It may be one of the class values in a partition in a non-hierarchical model. In this case it's value is relative, that is local to the non-hierarchical model. It may also be the class value returned from a top level partition in the hierarchy in which case it's value is absolute.

<TWOCCLASS> is a binary classification model. There are two versions of control file: one for training and one for classification using the trained model. The command, **multi_borders**, reads a training control file and outputs a classification control file. For training, <TWOCCLASS> contains a double-quoted set of parameters or options for training a two-class model. For classification, it is the name of a trained, binary classification model. The **multi_borders** command returns a series of statements for training each of the binary classifiers required for the over-all model, in addition to the final control file which contains the names of each.

Table 1: Summary of multi-borders validation results

Algorithm	train time (s)	class. time (s)	unc. coeff.	acc.	corr.	correlation cond. prob.
AGF	N/A	235	0.43	0.56	0.92	1.
Non-hierarchical	189	2.0	0.41	0.53	0.91	0.94
Hierarchical	111	0.84	0.42	0.54	0.91	0.89

The `classify_m` command takes the output from `multi_borders` and uses it to perform classifications on a set of test data. If the model has only one level, all the conditional probabilities are returned, otherwise only the winning probability is returned. Command-line programs use AGF with borders sampling (`class_borders`) as the binary classification model, however the source-level, C++ interface allows the user to specify any binary classification model desired.

4 Test scenarios

To test the algorithm we use some of the same satellite humidity data as described in Mills (2009). The specific humidity values are discretized into eight classes by dividing them at seven geometricly increasing values from 0.001 to 0.00007. Classes are labelled from 0 to 7 from lowest to highest humidity ranges. Two experiments were done. The first used non-hierarchical classification by partitioning the classes between each adjacent class, as shown in the following control file:

```
"" 0 / 1 2 3 4 5 6 7;
"" 0 1 / 2 3 4 5 6 7;
"" 0 1 2 / 3 4 5 6 7;
"" 0 1 2 3 / 4 5 6 7;
"" 0 1 2 3 4 / 5 6 7;
"" 0 1 2 3 4 5 / 6 7;
"" 0 1 2 3 4 5 6 / 7;
{0 1 2 3 4 5 6 7}
```

The blank options section means options can be passed from the command line. The second experiment was hierarchical and partitioned the classes recursively in half:

```

"-s 150 -W 40 -k 300" {
  "-s 100 -W 30 -k 250" {
    "-s 75 -W 25 -k 200" {0 1}
    "-s 75 -W 25 -k 200" {2 3}
  }
  "-s 100 -W 30 -k 250" {
    "-s 75 -W 25 -k 200" {4 5}
    "-s 75 -W 25 -k 200" {6 7}
  }
}

```

The results from this experiment are shown in Table 1, where they are compared with an AGF model with no pre-training. While accuracy suffers somewhat using the multi-borders models, there is an enormous improvement in classification speed, while training times are less than the classification times for the untrained model.

For the non-hierarchical model, conditional probabilities were solved using a simple linear least squares. Accuracy of estimates could likely be improved by using constraints or regularization (Press et al., 1992).

5 Conclusions

Software was described that allows one to specify, in a recursive and general way, a multi-class classification model comprised of one or more binary classifiers. The system was tested on discretized satellite humidity data using both a strictly hierarchical and strictly non-hierarchical model and compared with a direct kernel estimator without any pre-training. While the accuracy of both pre-trained models suffered somewhat compared to the classifier without pre-training, time performance was greatly improved. Software is available at: <http://libagf.sourceforge.net>.

References

- Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27.
- Mills, P. (2009). Isoline retrieval: An optimal method for validation of advected contours. *Computers & Geosciences*, 35(11):2020–2031.

- Mills, P. (2011). Efficient statistical classification of satellite measurements. *International Journal of Remote Sensing*, 32(21):6109–6132.
- Müller, K.-R., Mika, S., Rätsch, G., Tsuda, K., and Schölkopf, B. (2001). An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201.
- Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C*. Cambridge University Press, 2nd edition edition.
- Terrell, D. G. and Scott, D. W. (1992). Variable kernel density estimation. *Annals of Statistics*, 20:1236–1265.